What's New with DFSORT? (V2R1 and V2R2 Features)

Samuel Smith

IBM

March 4th, 2015

Session Number: 17105

ssmith@us.ibm.com

Agenda

- **V2R1**
  - Functional enhancements
    - Alphanumeric tests
    - PARSE enhancements
    - Symbol enhancements
    - Add string at end of VLR records
  - Improve DFSORT/DB2 Synergy
    - Improve how DFSORT interacts with DB2 Utilities
    - Enhancements benefit all users of DFSORT
  - Dynamic Sort Enhancements
    - Managing central storage usage
    - Control how DFSORT manages central storage usage
  - Support for 64-Bit Callers with 64-Bit Addressed Records
    - Invoke DFSORT from 64-bit address mode programs
    - Pass 64-bit addressed records to DFSORT with exits
- **V2R2**
  - Functional enhancements
    - Date conversion AGE function
    - Date conversion WEEKNUM function
  - Exploitation of zHPF
    - Update DFSORT to prefer BSAM
  - Message Updates
  - Joinkeys indicators in SMF data

V2R1 – Alphanumeric tests

- Perform comparison tests to see if a field only contains characters in a specific set (e.g. A-Z, a-z and/or 0-9)
  – Alphanumeric Tests: New UC, LC, MC, UN, LN and MN keywords (similar to the previously available NUM keyword) now allows you to test a field for various combinations of alphanumeric characters or non-alphanumeric characters using binary (BI) format.
    - **UC:** Uppercase characters (A-Z)
    - **LC:** Lowercase characters (a-z)
    - **MC:** Mixed case characters (A-Z, a-z)
    - **UN:** Uppercase and numeric characters (A-Z, 0-9)
    - **LN:** Lowercase and numeric characters (a-z, 0-9)
    - **MN:** Mixed case and numeric characters (A-Z, a-z, 0-9)

- This new support allows them to specify various sets of characters using a single compare condition.
  – Previously you would have had to code multiple compare conditions

V2R1 – Alphanumeric tests

- Alphanumeric Tests Usage:
  - You can use alphanumeric test keywords(UC, LC, MC, UN, LN and MN) in the following comparison operands: COND, INCLUDE, OMIT, BEGIN, END, WHEN and TRLID.

- Examples:

```
INCLUDE COND=(11,10,BI,EQ,MC)
OMIT COND=(50,5,BI,EQ,LC)
INREC IFTHEN=(WHEN=(1,3,BI,EQ,UC),OVERLAY=(20:C'UPPER'))
OUTREC IFTHEN=(WHEN=GROUP,BEGIN=(11,5,BI,EQ,UN),PUSH=(ID=8))
```

- PARSE function can now be used with alphanumeric test keywords(UC, LC, MC, UN, LN and MN) to start or end when a character from any of various alphanumeric character sets

- Example:

```
INREC PARSE=(%01=(ENDBEFR=UC,FIXLEN=5)),BUILD=(%01)
```

V2R1 – Parse enhancements

- Alphanumeric test Example:

Input data

```
----+----1----+----2
********************
$@$321%@%$FRANK
@053$SUSAN
%%%$$836%$VICKY
********************
```

Control card

```
INREC PARSE=(%01=(STARTAT=NUM,FIXLEN=3),
             %02=(STARTAT=UC,FIXLEN=10)),
BUILD=(%01,X,%02)
SORT FIELDS=(1,3,CH,A)
```

Output data

```
----+----1----+
***************
053 SUSAN
321 FRANK
836 VICKY
***************
```

V2R1 – Parse enhancements

- Parse a very large number of delimited fields and records with consecutive fields that need to be parsed in the same way. Now you can parse fields based on a specific set of characters.

- PARSE Enhancements:
  - **REPEAT=v** is a new PARSE option that can be used to repeat a particular parse field definition multiple times.
  - **STARTAFT=an**, **STARTAT=an**, **ENDBEFR=an** and **ENDAT=an** can now be used with the PARSE function to start or end when a character from any of various alphanumeric character sets is found.

- Previously you could on test for strings or blanks

- PARSE various sets of characters using a single PARSE keyword

V2R1 – Parse enhancements

- More Parsed Fields:
- You can now use up to 1000 parsed fields (%0-%999) with the PARSE function; the previous limit was 100 parsed fields (%0-%99).
- Example:

```
OUTREC PARSE=(%121=(ENDBEFR=C',',FIXLEN=12),
              %322=(ENDBEFR=C',',FIXLEN=8),
              %999=(FIXLEN=5)),
       BUILD=(%121,X,%322,X,%999)
```

- **REPEAT=v** can be used with %n, %nn or %nnn to specify v identically defined consecutive parsed fields for which data is to be extracted. The parsed fields will start with the %n, %nn or %nnn field you select and be incremented by one for each repeated parsed field.
- Example:

```
INREC PARSE=(%=(ENDBEFR=C',',REPEAT=3),
             %11=(ENDBEFR=C',',FIXLEN=10,REPEAT=4)),
      BUILD=(%11,X,%12,X,%13,X,%14)
```

V2R1 – Parse enhancements

- Use of REPEAT Example:

Input data

```
----+----1----+----2----+----3----+----4----+----5----+
*******************************************************
A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
*******************************************************
```

Control card

```
----+----1----+----2----+----3----+----4----+----5----+
*******************************************************
OPTION COPY
INREC PARSE=(%01=(ENDBEFR=C',',FIXLEN=1,REPEAT=26)),
BUILD=(%16)
*******************************************************
```

Output data

```
----+----1
**********
P
**********
```

V2R1 – Parse enhancements

- Example extracting the last node:

Input data

```
----+----1----+----2----+----3----+
***********************************
SYS1.PARMLIB
SYS2.NPLEX1.FILE.NODE4
COMP1.WEEK1.DAY01.SESSION1
SYS1.GDG.DATASET5.TEST.G0005V00
***********************************
```

Control card

```
----+----1----+----2----+----3----+----4----+----5
**************************************************
OPTION COPY
INREC IFOUTLEN=8,
IFTHEN=(WHEN=INIT,PARSE=(%=(ENDAT=C' '),
%01=(SUBPOS=9,FIXLEN=8)),BUILD=(%01)),
IFTHEN=(WHEN=(1,8,SS,EQ,C'.'),
PARSE=(%02=(STARTAFT=C'.',FIXLEN=8)),BUILD=(%02))
**************************************************
```

Output data

```
----+----1----+
***************
PARMLIB
NODE4
SESSION1
G0005V00
***************
```

V2R1 – Symbol enhancements

- DFSORT symbols are now supported for more DFSORT operands, especially those of the form KEYWORD=n.

- Symbol Enhancements:
  - **KEYWORD=sym** will be supported for operands of the form KEYWORD=n where n is a number.

- This new support allows them to use symbols for these types of keywords. Symbols can now be used with more DFSORT features such as **ID=sym**, **SEQ=sym**, **ABSPOS=sym** and **FIXLEN=sym**.

- Symbols make ease of coding is almost like writing a sentence

V2R1 – Symbol enhancements

- Symbol Enhancements:
  - **KEYWORD=n** can now be specified as KEYWORD=symbol where symbol represents an equivalent number (for example, if you have New_Length,25 in SYMNAMES, you can use LENGTH=New_Length wherever you can use LENGTH=25).
  - The operands ABSPOS, ACCEPT, ADDPOS, AVGRLEN, DO, ENDPOS, ENDREC, FIXLEN, ID, IFOUTLEN, INCR, LENGTH, LINES, MAXLEN, RECORDS, REPEAT, SAMPLE, SEQ, SKIPREC, SPLIT1R, SPLITBY, START, STARTPOS, STARTREC, STOPAFT and SUBPOS can now have symbols.

- Example:

```
OUTFIL REPEAT=Mult,
       IFOUTLEN=out_length,
       IFTHEN=(WHEN=GROUP,RECORDS=Num_records,
               PUSH=(id_col:ID=id_length))
```

V2R1 – Symbol enhancements

- Example:

Symnames DD

```
----+----1----+----2
********************
//SYMNAMES DD *
RPT-CNT,20
IFLEN,50
TEMP-REC,81,11,CH
TEMP-DTE,=,8,CH
SKIP,1
TEMP-SEQ,*,2,ZD
DOVAL,1
********************
```

Control card

```
----+----1----+----2----+----3----+----4----+----5----+----6
************************************************************
OUTFIL REPEAT=RPT-CNT,
IFTHEN=(WHEN=INIT,BUILD=(TEMP-REC:DATE1,X,SEQNUM,2,ZD)),
IFTHEN=(WHEN=INIT,OVERLAY=(001:81,8,Y4T(-),31:TEMP-SEQ)),
IFTHEN=(WHEN=INIT,FINDREP=(INOUT=(C'-',C'/'),DO=DOVAL))
************************************************************
```

```
----+----1----+----2----+----3----+----4----+----5.....8----+----9----+
------- ORIGINAL STATEMENTS FROM SYMNAMES -------
RPT-CNT,20
IFLEN,50
TEMP-REC,81,11,CH
TEMP-DTE,=,8,CH
SKIP,1
TEMP-SEQ,*,2,ZD
DOVAL,1
------------------ SYMBOL TABLE -----------------
RPT-CNT,20
IFLEN,50
TEMP-REC,81,11,CH
TEMP-DTE,81,8,CH
TEMP-SEQ,90,2,ZD
DOVAL,1
2015/02-06                    01                    .....  20150206 01
2015/02-06                    02                    .....  20150206 02
...                           ...                          ...
2015/02-06                    19                    .....  20150206 19
2015/02-06                    20                    .....  20150206 20
**********************************************************************
```

Output data

- A SYMBOL can be a constant, a field or a parsed field

V2R1 – Add a string at the end of VLR records

- Insert a string at the end of a variable-length record

- Add string at end of VLR records:
  – Add specific characters at the end of each VB record, such as X’0D0A’ (CRLF).

- Allows you to add a string of 1-50 characters at the end of each VB record.
  – Previously E35 exit logic would be required to add a string to each VB record.

V2R1 – Add a string at the end of VLR records

- Adding a string at the end of Variable-Length Records:

- **VLTRAIL=string** is a new OUTFIL option that allows you to insert a character string
  - Insert a character string using **(C'string')**
  - Insert a hexadecimal string using **(X'yy...yy')**
  - Added to the end of each variable-length OUTFIL output record

- Example:

  OUTFIL VLTRIM=C' ',VLTRAIL=X'0D0A'

V2R1 – Add a string at the end of VLR records

- Example:

Input data

```
----+----1
**********
A
AB
ABC
ABCD
ABCDE
ABCDEF
ABCDEFG
ABCDEFGH
**********
```

Control card

```
----+----1----+----2----+----3----+----4----+----5----+-
********************************************************
SORT FIELDS=(1,20,CH,D)
OUTFIL FTOV,VLTRIM=C' ',VLTRAIL=C' @SHARE',
HEADER1=('THIS IS A TRANSMITTED FILE',/,
         'THIS IS MY SECOND LINE HEADER'),
TRAILER1=('NUMBER OF RECORDS : ',COUNT+3=(M11,LENGTH=3))
********************************************************
```

Output data

```
----+----1----+----2----+----3----+--
*************************************
THIS IS A TRANSMITTED FILE @SHARE
THIS IS MY SECOND LINE HEADER @SHARE
ABCDEFGH @SHARE
ABCDEFG @SHARE
ABCDEF @SHARE
ABCDE @SHARE
ABCD @SHARE
ABC @SHARE
AB @SHARE
A @SHARE
NUMBER OF RECORDS : 011 @SHARE
*************************************
```

V2R1 – DFSORT/DB2 interaction

- Improve reliability and performance of DFSORT when invoked by DB2 Utilities
  - Provide virtual storage constraint relief below 16MB
  - Reduce disk work space related failures
  - Improve scalability for very large sorts
- Exploit Extended TIOT, uncaptured UCBs and above the line DSAB options for dynamically allocated work data sets
  - Expand “additional” work data sets capability provided in previous release
  - Increase maximum size of disk and memory object work files
- Improved reliability and scalability for all users of DFSORT

V2R1 – DFSORT/DB2 interaction

- Extended TIOT
  - DFSORT V1R12 provided support for programs that invoke DFSORT, ICETOOL or ICEGENER and dynamically allocate input, output and work data sets using the options for Extended TIOT, uncaptured UCBs, and DSAB above 16 megabyte virtual.

- In DFSORT V2R1, these options are exploited by DFSORT's dynamic allocation of work data sets
  - Uncaptured UCBs (S99UCACB option) is always exploited
  - Extended TIOT (S99TIOEX option)
  - DSAB above 16 megabyte virtual (S99DSAB option) are used if DFSORT is running authorized

- The exploitation is automatic

V2R1 – Dynamic sort enhancements

- Additional work data sets
  - DFSORT V1R12 provided capability for dynamic allocation of additional work data sets that are only used if needed
  - Primary space of zero
  - Secondary space only allocated if needed
- DFSORT V2R1 now provides capability to provide similar function for JCL or preallocated work data sets.
  - Work data sets with primary allocation of zero are only used when work data sets with non-zero primary have been exhausted
- Example:

```
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(0,5))
```

- SORTWK03 is only used if no more space can be allocated to SORTWK01 and SORTWK02

V2R1 – Dynamic sort enhancements

- Increased size for work files
  - Further exploit Extended Address Volumes(EAV)
  - Maximum number of tracks on a single work data increased from 1,048,576 to 16,777,216 when full track blocking is used

- In cases where DFSORT is unable to use full track blocking, the limit may be less

- To further exploit large central storage configurations
  - Maximum amount of Memory object storage that can be used as intermediate work space has been increased from 64 gigabytes to 1 terabyte

V2R1 – Dynamic sort enhancements

- Dynamic Sort Enhancements
  - Better management over DFSORT's use of central storage
  - Avoid over commitment of central storage and paging resources
  - Decrease unbalanced use of resources used by concurrent sort applications
- Improved DFSORT installation defaults to control DFSORT's use of central storage
  - New **TUNE** installation default to favor storage or disk work space optimization
    - Installation defaults can be specified as a percentage of current resources instead of a percentage of configured storage
    - Allocation of central storage in smaller increments
- Tailor DFSORT's use of central storage usage based on system configuration and requirements
- Gradual allocation in smaller increments allows for checking available resources and avoiding over commitment

V2R1 – Dynamic sort enhancements

- New TUNE installation default
- **TUNE=STOR** (shipped default)
  - DFSORT is to allocate storage in smaller increments
  - Increases allocation for dynamically allocated work data sets to reduce risk of failure if storage becomes constrained
- **TUNE=DISK**
  - DFSORT allocates all storage to be used immediately
  - Reduces allocation for dynamically allocated work data sets based on storage already obtained
- **TUNE=OLD**
  - DFSORT allocates storage as it did in prior release
- **EXPOLD** shipped default changed from **MAX to 50%**
  - Controls amount of storage in use by other applications (“old frames”) that DFSORT considers available for sorting
  - With TUNE=STOR, do not allow a sort to obtain more than 50% of old frames dynamically calculated at run time - If TUNE=OLD, then calculation is 50% of configured storage
- **EXPRES** shipped default changed from **0 to 10%**
  - Controls storage (available + old) that DFSORT considers unavailable for sorting applications
  - With TUNE=STOR, the reserved amount will not fall below 10% of available frames
  - If TUNE=OLD, then calculation is 10% of configured storage

V2R1 – Dynamic sort enhancements

- Coexistence Considerations
  - Some sort applications may use less central storage in an effort to reduce impact to overall system performance
  - Reduced central storage usage may lead to an increase in disk work space usage and/or an increase in elapsed time

- To force DFSORT to operate as it did in V1R12(K00) set the following installation defaults:
  - TUNE=OLD
  - EXPOLD=MAX
  - EXPRES=0

- IBM DFSORT does not recommend running with the old defaults without consulting the IBM support center

V2R1 – Support for 64bit callers

- Support for 64-Bit Callers with 64-Bit Addressed Records
  - Increase use of “above bar” storage where “above bar” is defined as an address greater than a 2-gigabyte address.

- Eligible user programs and exits can now be written to:
  - Call DFSORT from a program in 64-bit address mode (AMODE 64)
  - Use DFSORT E15, E35 and E32 exits running in 64-bit address mode (AMODE 64)
  - Pass 64-bit addressed records to DFSORT using E15, E32, and E35 exits

- Less use of below bar storage and increased capacity of area to store records before exits pass records to DFSORT

V2R1 – Support for 64bit callers

- Invocation
  - Call DFSORT from a program in 64-bit addressing mode (AMODE 64) using a new 64-bit invocation parameter list and the entry name ICEMAN64 or SORT64
  - The invoking program uses the new 64-bit parameter list and must use ICEMAN64 or SORT64 as the entry point name for the LINK, ATTACH or XCTL macro
  - A 64-bit Register 1 points at the new 64-bit invocation parameter list
  - You can still pass DFSORT a 31bit or 24bit parameter list
    - Our recommendation is to run with the new 64bit parameter list

V2R1 – Support for 64bit callers

- 64-bit Invocation Parameter List (Additional Detail Continued)
  - Reserved fields are defined and must be set to zero as defined
  - When a 32-bit exit parameter list is used, a 32-bit Register 1 is returned from exit, and exit saves 32-bit registers in save area (Format-0 72 byte save area) pointed at by 32-bit Register 13
  - When a 32-bit Register 1 is returned, the high half of Register 1 will be ignored by DFSORT when the exit returns to DFSORT
  - When a 64-bit exit parameter list is used, a 64-bit Register 1 is returned from exit, and exit saves 64-bit registers in save area (Format-4 144 byte save area) pointed at by 64-bit Register 13
  - Address of exit must be a 'clean' 31-bit address or a 'clean' 24-bit address
  - Address of control statement area or ALTSEQ Translation table can be above or below the bar

V2R1 – Support for 64bit callers

- 64-bit E15 Exit parameter list

- DFSORT passes to the E15 exit a 64-bit general register 1 with an address of a parameter list that contains the record address and the user address constant
- The format of the parameter list is:
  - Bytes 1 through 4 X'00000000' Address of the new record
  - Bytes 5 through 8 X'00000000' User exit address constant

- DFSORT provides a 144-byte Format 4 save area pointed to by 64-bit register 13 in which the exit can save the 64-bit registers.
- The E15 should expect and use 64-bit addresses
- Before returning control to DFSORT, you must:
  - Place the address of the record in a 64-bit register, a 64-bit address
  - Must be a clean 31-bit address or a clean 24-bit address.
  - Put the return code in 64-bit register 15.

V2R1 – Support for 64bit callers

- 64-bit E35 Exit Parameter List
- DFSORT passes to the E35 exit a 64-bit general register 1 with an address of a parameter list that contains the 2 record addresses and the user address constant.
- The format of the parameter list is:
  - Bytes 1 through 8 X'00000000' Address of record leaving DFSORT
  - Bytes 9 through 16 X'00000000' Address of record in output area
  - Bytes 17 through 24 X'00000000' User exit address constant

- DFSORT provides a 144-byte Format 4 save area pointed to by 64-bit register 13 in which the exit can save the 64-bit registers.
- The E35 should expect and use 64-bit addresses.
- Before returning control to DFSORT, you must:
  - Place the address of the record in 64-bit register 1. This must be a 64-bit address, a clean 31-bit address or a clean 24-bit address.
  - Put the return code in 64-bit register 15.

V2R1 – Support for 64bit callers

- 64-bit E32 Exit Parameter List
- DFSORT passes to the E32 exit a 64-bit general register 1 with an address of a parameter list that contains the record address and the user address constant.
- The format of the parameter list is:
  - Bytes 1 through 8 X'00000000' Increment of next file to be used for input
  - Bytes 9 through 16 X'00000000' address of next input record
  - Bytes 17 through 24 X'00000000' User exit address constant
- DFSORT provides a 144-byte Format 4 save area pointed to by 64-bit register 13 in which the exit can save the 64-bit registers.
- The E32 should expect and use 64-bit addresses.
- Before returning control to DFSORT, you must:
  - Place the address of the next input record from the requested input file in the second doubleword of the parameter list. This must be a 64-bit address, a clean 31-bit address or a clean 24-bit address.
  - Put the return code in 64-bit register 15.

V2R1 – Support for 64bit callers

- MODS Control Statement New N64 Parameter

- When **N64** is specified:
  - This indicates that your routine (exit) will use a 64-bit exit parameter list
  - It has already been bound or link-edited and can be used by DFSORT without further binding or link-editing

- 64-bit Invocation Parameter List (Additional Detail)
  - The invoking program uses the new 64-bit parameter list and must use ICEMAN64 or SORT64 as the entry point name for the LINK, ATTACH or XCTL macro.
  - A 64-bit Register 1 points at the new 64-bit invocation parameter list shown in the next slide

MODS Exx=(Exit Name,Storage in Bytes,DDname for Exit Loadlib,N64)

V2R1 – Support for 64bit callers

- ICEPL64 DSECT Contents

```
****************************************************************
* 64-Bit Invocation Parameter List for DFSORT
****************************************************************
ICE64INV DSECT 64-Bit Invocation Parameter List
ICEPLID DC C'PL64SORT' Identifier
ICEMDEX1 DS C Flag byte 1
ICE15A24 EQU X'80' AMODE 24 for E15 Exit or
ICE32A24 EQU X'80' AMODE 24 for E32 Exit
ICE15A31 EQU X'40' AMODE 31 for E15 Exit or
ICE32A31 EQU X'40' AMODE 31 for E32 Exit
ICE15A64 EQU X'20' AMODE 64 for E15 Exit or
ICE32A64 EQU X'20' AMODE 64 for E32 Exit
ICE35A24 EQU X'10' AMODE 24 for E35 Exit
ICE35A31 EQU X'08' AMODE 31 for E35 Exit
ICE35A64 EQU X'04' AMODE 64 for E35 Exit
ICE18A24 EQU X'02' AMODE 24 for E18 Exit
ICE18A31 EQU X'01' AMODE 31 for E18 Exit
ICEMDEX2 DS C Flag byte 2
ICERSVD1 EQU X'80' Reserved - must be set to zero
ICE39A24 EQU X'40' AMODE 24 for E39
ICE39A31 EQU X'20' AMODE 31 for E39
ICERSVD2 EQU X'10' Reserved - must be set to zero
```

V2R1 – Support for 64bit callers

- ICEPL64 DSECT Contents(Continued)

```
*****************************************************************
* 64-Bit E15 Exit Parameter List for DFSORT
*****************************************************************
ICE64E15 DSECT 64-Bit E15 Exit Parameter List
ICE15NR DS D X'00000000' | Address of the new record
ICE15UC DS D X'00000000' | User exit address constant
*
*****************************************************************
* 64-bit E32 Exit Parameter List for DFSORT
*****************************************************************
ICE64E32 DSECT 64-Bit E32 Exit Parameter List
ICE32NXF DS D X'00000000' | Increment of next file to be used
ICE32NIR DS D X'00000000' | Address of next input record from E32
ICE32UC DS D X'00000000'  | User exit address constant
*
*****************************************************************
* 64-bit E35 Exit Parameter list for DFSORT
*****************************************************************
ICE64E35 DSECT 64-Bit E35 Exit Parameter List
ICE35RL DS D X'00000000' | Address of record leaving DFSORT
ICE35RO DS D X'00000000' | Address of record in output area
ICE35UC DS D X'00000000' | User exit address constant
```

V2R1 – Support for 64bit callers

- 64-bit Invocation Parameter List (Additional Detail)

```
Hex Dec Len Bit Contents
0 0 8 C'PL64SORT'
8 8 1 0 AMODE 24 E15 or E32
      1 AMODE 31 E15 or E32
      2 AMODE 64 E15 or E32
      3 AMODE 24 E35
      4 AMODE 31 E35
      5 AMODE 64 E35
      6 AMODE 24 E18
      7 AMODE 31 E18
9 9 1 0 Reserved. Must be set to 0.
      1 AMODE 24 E39
      2 AMODE 31 E39
      3 Reserved, Must be set to 0.
      4 E15 or E32 type of exit parameter list used and how Register 1 should be interpreted
      5 E35 type of exit parameter list used and how Register 1 should be interpreted
      6 E18 type of exit parameter list used and how Register 1 should be interpreted
      7 E39 type of exit parameter list used and how Register 1 should be interpreted
A 10 14 Reserved. Must be set to zeroes.
18 24 8 Address of control statement area (zeros if none)
20 32 8 X'00000000' | Address of user exit E15 or E32 (zeros if none)
28 40 8 X'00000000' | Address of user exit E35 (zeros if none)
30 48 8 X'00000000' | Address of user exit constant
38 56 8 Address of ALTSEQ Translation table (zeros if none)
40 64 8 X'00000000' | Address of ESTAE Area Pointer (zeros if none)
48 72 8 X'00000000' | Address of user exit E18 (zeros if none)
50 80 8 X'00000000' | Address of user exit E39 (zeros if none)
50 88 8 X'00000000' | 4-Character call identifier (zeros if none)
60 96 40 Reserved. Must be set to zeros.
```

V2R1 – Support for 64bit callers

- New Error Message ICE290A

- **ICE290A INVALID 64-BIT INVOCATION PARAMETER LIST - REASON CODE IS rsn**

  – DFSORT was invoked with a 64-bit invocation parameter list. However, an error was found in this parameter list. Reason code values (rsn) are indicated. DFSORT terminates.

  – Reason code (rsn) values and programmer responses are listed in the DFSORT Messages and Codes publication

  – **System Action:** DFSORT terminates

V2R1 – Support for 64bit callers

- New Informational Message ICE291I

- **ICE291I AMODE FLAG FOR EXIT WITHOUT CORRESPONDING EXIT ADDRESS IGNORED**
  - An AMODE flag was set for an exit for which a corresponding exit address was not present in the parameter list. For example, the AMODE 64 flag was set for the E15 exit, but the E15 exit address was zeros in the 64-bit invocation parameter list.
  - **System action:** Processing continues. Each AMODE flag for an exit without a corresponding exit address is ignored.
  - **Programmer response:** No action is necessary. If you want to eliminate this warning message, turn off the unneeded AMODE flag or specify a corresponding exit address.

V2R1 – Support for 64bit callers

- Updated Error Message ICE034A

- **ICE034A MODS STATEMENT OPERAND ERROR**

  – New reason code was added for the case when MODS N64 parameter is used incorrectly.

  – N64 was specified for the fourth parameter, but DFSORT was not invoked using a 64-bit invocation parameter list.

  – **System action:** The program terminates.

  – **Programmer response:** N64 is not specified as the fourth parameter if DFSORT was not invoked using a 64-bit invocation parameter list.

V2R2 – Date conversion AGE function

- Calculate the date duration that specifies the number of years, months, and days between an input date and current date.

- A date conversion function AGE for the BUILD and OVERLAY operands of DFSORT's INREC, OUTREC and OUTFIL statements can now be used to calculate the date duration in three different forms.
  - **AGE=YMD** produces a 8 byte result which has duration in years (0-9999), months (00-12), and days (00-31).
  - **AGE=YM** produces a 6 byte result which has duration in years (0-9999), months (00-12).
  - **AGE=YD** produces a 7 byte result which has duration in years (0-9999), days (00-366).

- This new support allows the users to perform date conversion operations and calculate the age.
  - Previously, they would have had to code a program to achieve the same results.

V2R2 – Date conversion AGE function

- Usage of AGE function:

- You can use AGE function on the BUILD and OVERLAY operands of DFSORT's INREC, OUTREC and OUTFIL statements.

- Examples:

  INREC OVERLAY=(35:01,7,Y4T,AGE=**YMD**)

  OUTREC BUILD=(31,8,Y4W,AGE=**YM**)

  OUTFIL OVERLAY=(64:52,4,Y4U,AGE=**YD**)

  - Assuming the current day is February 12th, 2015

V2R2 – Date conversion AGE function

- AGE =YD - Example:

Input data

```
----+----1----+----2----+----3----+
***********************************
20001231   - DEC 31 2000
19820312   - MAR 12 1982
17760704   - USA INDEPENDENCE DAY
***********************************
```

Control card

```
----+----1----+----2----+----3----+
***********************************
OPTION COPY
INREC OVERLAY=(35:1,8,Y4T,AGE=YD)
***********************************
```

Output data

```
----+----1----+----2----+----3----+----4-
*****************************************
20001231   - DEC 31 2000           0014037
19820312   - MAR 12 1982           0032331
17760704   - USA INDEPENDENCE DAY  0238217
*****************************************
```

V2R2 – Date conversion AGE function

- AGE=YM – Example:

Input data

```
----+----1----+----2----+----3----+----4
****************************************
                              3662000
                              0601982
                              3451944
 INVALID DATE                 7001944
****************************************
```

Control card

```
----+----1----+----2----+----3----+
***********************************
OPTION COPY
OUTREC BUILD=(31,7,Y4W,AGE=YM)
***********************************
```

Output data

```
----+----1
**********
001401
003211
007001
******
**********
```

- **ICE288I 0 INPUT OR OUTPUT DATE VALUE OUT OF RANGE FOR DATE CONVERSION OR DATE ARITHMETIC**

V2R2 – Date conversion AGE function

- AGE=YMD Example:

Input data

```
----+----1----+----2----+----3----+
***********************************
20001231  - DEC 31 2000
19820312  - MAR 12 1982
17760704  - USA INDEPENDENCE DAY
***********************************
```

Control card

```
----+----1----+----2----+----3----+
***********************************
OPTION COPY
INREC OVERLAY=(35:1,8,Y4T,AGE=YMD)
***********************************
```

Output data

```
----+----1----+----2----+----3----+----4----+
*********************************************
20001231  - DEC 31 2000           00140106
19820312  - MAR 12 1982           00321025
17760704  - USA INDEPENDENCE DAY  02380702
*********************************************
```

V2R2 – Date conversion WEEKNUM function

- Calculate the week of the year for a specific input date.
- A date conversion function WEEKNUM for the BUILD and OVERLAY operands of DFSORT's INREC, OUTREC and OUTFIL statements can now be used to calculate the week number that represents the week of the year.
  - **WEEKNUM=USA** returns an integer in the range of 1 to 54 that represents the week of the year.
    - The week starts with Sunday, and January 1 is always in the first week.
  - **WEEKNUM=ISO** function returns an integer in the range of 1 to 53 that represents the week of the year.
    - The week starts with Monday and includes 7 days.
- This new support allows the users to perform date conversion operations and calculate the week number.
- Previously, you would have had to code a program to achieve the same results.

V2R2 – Date conversion WEEKNUM function

- Usage of WEEKNUM function

- You can use WEEKNUM function on the BUILD and OVERLAY operands of DFSORT's INREC, OUTREC and OUTFIL statements.
- Examples:

INREC BUILD=(1,45,10,8,Y4T,WEEKNUM=**USA**,46,300)

OUTREC OVERLAY=(28:66,5,Y2W,WEEKNUM=**ISO**)

OUTFIL BUILD=(30,55,76:1,5,Y4V,WEEKNUM=**USA**)

Input-Date	WEEKUM=USA	WEEKNUM=ISO
2000-01-01	01	52
2000-01-02	02	52
2000-01-03	02	01
2014-01-01	01	01
2014-01-02	01	01
2014-06-09	24	24
2014-12-29	53	01
2014-12-30	53	01
2014-12-31	53	01

V2R2 – Date conversion WEEKNUM function

- WEEKNUM=ISO Example:

Input data

```
----+----1----+
***************
2000-01-01
2000-01-02
2000-01-03
2000-12-31
2014-01-02
2014-06-09
2014-12-31
***************
```

Control card

```
----+----1----+----2----+----3----+-----+
*****************************************
OPTION COPY
INREC OVERLAY=(15:1,10,UFF,M11,LENGTH=8,
               30:15,8,Y4T,TOJUL=Y4W)
OUTREC BUILD=(1,40,30,7,Y4W,WEEKNUM=ISO)
*****************************************
```

Output data

```
----+----1----+----2----+----3----+----4----+
*********************************************
2000-01-01    20000101       0012000    52
2000-01-02    20000102       0022000    52
2000-01-03    20000103       0032000    01
2000-12-31    20001231       3662000    52
2014-01-02    20140102       0022014    01
2014-06-09    20140609       1602014    24
2014-12-31    20141231       3652014    01
*********************************************
```

- WEEKNUM=ISO function returns an integer in the range of 1 to 53 that represents the week of the year. The week starts with Monday and includes 7 days. Week 1 is the first week of the year to contain a Thursday, which is equivalent to the first week containing January 4. Thus, it is possible to have up to 3 days at the beginning of the year appear as the last week of the previous year, or to have up to 3 days at the end of a year appear as the first week of the next year.

V2R2 – Date conversion WEEKNUM function

- WEEKNUM=USA Example:

Input data

```
----+----1----+
***************
2000-01-01
2000-01-02
2000-01-03
2000-12-31
2014-01-02
2014-06-09
2014-12-31
***************
```

Control card

```
----+----1----+----2----+----3----+----4----+
*********************************************
OPTION COPY
INREC OVERLAY=(15:1,10,UFF,M11,LENGTH=8)
OUTREC BUILD=(1,29,15,8,Y4T,WEEKNUM=USA)
*********************************************
```

Output data

```
----+----1----+----2----+----3----+
***********************************
2000-01-01    20000101       01
2000-01-02    20000102       02
2000-01-03    20000103       02
2000-12-31    20001231       54
2014-01-02    20140102       01
2014-06-09    20140609       24
2014-12-31    20141231       53
***********************************
```

V2R2 – Update DFSORT to use BSAM

- Improved DFSORT performance and exploitation of zHPF

- High Performance FICON for System z (zHPF) is a data transfer protocol that is optionally employed for accessing data from IBM DS8000 storage and other subsystems.

  - DFSORT normally uses EXCP for processing of basic and large format sequential input and output data sets (SORTIN, SORTOUT, OUTFIL).
  - DFSORT already uses BSAM for extended format sequential input and output data sets (SORTIN, SORTOUT and OUTFIL).
  - DFSORT will be updated to prefer the use of BSAM for SORTIN, SORTOUT, and OUTFIL when zHPF is available.

- This support of zHPF allows for the new System z I/O architecture, whose channel programs allow to reduce elapsed time and increase I/O rates (up to 2x).

V2R2 – Update DFSORT to use BSAM

- zHPF Usage:
- DFSORT will automatically take advantage of zHPF if it is available on your system.
- No user actions are necessary
- Software Dependencies
  - None
- Hardware Dependencies
  - Presence of High Performance Ficon (HPF) hardware

V2R2 – Message update

- Updated error message ICE099A

- Message ICE099A issued after BLDL failed contained DD name for data set but did not have a member name.

- The Message ICE099A was changed from:
  - ICE099A BLDL FAILED FOR (dd name) DATA SET
- To:
  - ICE099A BLDL FAILED FOR (dd name) DATA SET, MEMBER (member name)

- The programmer response instructs the user to verify that the member exists in the dataset.
- Providing the member name will simplify this process.

V2R2 – Joinkeys indicators in SMF data

- Better method for identifying the Joinkeys jobs

- Update ICESMF mapping macro to reference fields within the SMF type-16 record with Joinkeys job indicators.
  - ICEJOINM – Indicates Joinkeys Main Task
  - ICEJOIN1 – Indicates Joinkeys Sub Task1
  - ICEJOIN2 – Indicates Joinkeys Sub Task2

- Clients now can run reports to analyze the usage of Joinkeys jobs.

Questions?

Complete your session evalı

Appendix

- **Publications(V2R1)**
  - z/OS: DFSORT Installation and Customization (SC23-6881-00)
  - z/OS: DFSORT Tuning Guide (SC23-6882-00)
  - z/OS DFSORT Application Programming Guide (SC23-6878-00)
  - z/OS DFSORT Messages and Codes (SC23-6879-00)

- **Publications(V2R2)**
  - z/OS DFSORT Application Programming Guide (SC23-6878-01)
  - z/OS DFSORT Messages and Codes (SC23-6879-01)
  - z/OS DFSORT: Getting Started (SC23-6880-01)
  - z/OS: DFSORT Installation and Customization (SC23-6881-01)
  - z/OS: DFSORT Tuning Guide (SC23-6882-01)

- Web site: http://www.ibm.com/storage/dfsort

- Contacts:
  - DFSORT Hotline - dfsort@us.ibm.com

System z Social Media Channels

- Top Facebook pages related to System z:
  - IBM System z
  - IBM Academic Initiative System z
  - IBM Master the Mainframe Contest
  - IBM Destination z
  - Millennial Mainframer
  - IBM Smarter Computing
- Top LinkedIn groups related to System z:
  - System z Advocates
  - SAP on System z
  - IBM Mainframe- Unofficial Group
  - IBM System z Events
  - Mainframe Experts Network
  - System z Linux
  - Enterprise Systems
  - Mainframe Security Gurus
- Twitter profiles related to System z:
  - IBM System z
  - IBM System z Events
  - IBM DB2 on System z
  - Millennial Mainframer
  - Destination z
  - IBM Smarter Computing
- YouTube accounts related to System z:
  - IBM System z
  - Destination z
  - IBM Smarter Computing

- Top System z blogs to check out:
  - Mainframe Insights
  - Smarter Computing
  - Millennial Mainframer
  - Mainframe & Hybrid Computing
  - The Mainframe Blog
  - Mainframe Watch Belgium
  - Mainframe Update
  - Enterprise Systems Media Blog
  - Dancing Dinosaur
  - DB2 for z/OS
  - IBM Destination z
  - DB2utor

Trademarks

The following are trademarks of the International Business machines Corporation in the United States and/or other countries.

AIX*	DB2*	DFSORT	IBM*	Language Environment*	Redbooks*	System Storage	System z10 Business Class	z10 EC
BladeCenter*	DFSMS	Domino*	IBM eServer	MVS	REXX	System x*	Tivoli*	zEnterprise*
BookManager*	DFSMSdss	DS6000	IBM logo*	Parallel Sysplex*	RMF	System z	WebSphere*	zSeries*
CICS*	DFSMShsm	DS8000*	IMS	ProductPac*	ServerPac*	System z9	z10	
DataPower*	DFSMSrmm	FICON*	InfinBand	RACF*	SYSREXX	System z10	z10 BC	

- Registered trademarks of IBM corporation

The following are registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries. IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce. Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Windows Server and the Windows logo are trademarks of the Microsoft group of countries. TIL is a registered trademark, and a registered community trademark of the Office of Government Commerce,and is registered in the U.S. Patent and Trademark Office. UNIX is a registered trademark of The Open Group in the United States and other countries. Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom. Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

- Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography. This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.